SoftMAC – Flexible Wireless Research Platform

Michael Neufeld, Jeff Fifield, Christian Doerr, Anmol Sheth and Dirk Grunwald
Dept. of Computer Science
University of Colorado, Boulder
Boulder, CO 80309-0430

November 4, 2005

Abstract

Discontent with the traditional network protocol stack architecture has been steadily increasing as new network technologies and applications demand a greater degree of flexibility and "end to end" information than a strictly layered structure permits. *Software-defined radio* (SDR) systems have prompted some rethinking of the network stack, and schemes have been presented that attempt to take advantage of the opportunities afforded by these flexible radio systems. However, evaluation of these schemes has largely taken place only in simulation or on small testbeds. In order to seriously evaluate the utility and efficacy of architectures and heuristics that take advantage of SDR systems it is essential to construct and test these systems "in the wild," *i.e.* in environments outside of simulators and the laboratory. Unfortunately the SDR platforms currently available are not generally suitable for such deployments.

We argue that network researchers should begin to design for software defined radio systems and move from simulation to prototypes. To this end, we have developed a software system that allows researchers to use inexpensive, commodity wireless networking cards to experiment with new MAC layer protocols. Individual radios still use the common 802.11a/b/g RF modulations, but by judicious control of the interface we are able to completely supplant the standard 802.11 MAC. On top of this radio subsystem, we have created a software architecture that permits researchers to easily construct and deploy experimental dynamic MAC layers on systems running Linux. We argue by example that such simple software defined radios provide robust, capable platforms for research. Using such inexpensive tools will allow wireless network research to address a number of the opportunities of software defined radio.

1 Introduction

Computer systems researchers have a rich history of experimenting with wireless networks. Alohanet was one of the earliest wireless data networks, and the design of the collision avoidance protocols for that network influenced the design of Ethernet. Contributions by researchers interested in *packet radio* have greatly influenced protocol designs of existing products. For example, the RTS/CTS mechanism used in the 802.11 networking standard was influenced by the MACA protocol proposed by Phil Karn [1].

Recently, most of the experimentation in wireless networking has used commodity components such as 802.11b networking cards. This has occurred for a number of reasons – these commodity networking cards are inexpensive, do not require a license to operate, and offer good performance. Furthermore there are a wealth of interesting computer systems problems facing wireless networking that occur above the physical and MAC layers. Using these commodity cards allows researchers to build systems to investigate these problems.

However, recent research has shown that there is still "plenty of room at the bottom". While 802.11 has demonstrated sufficient utility for widespread adoption, there are still many interesting problems to solve at the MAC layer and below. For example, the 802.11 MAC protocol suffers from many weaknesses, opening it to attacks limiting service [2] or violating privacy. Protocols designed to overcome these security problems have also been found to have security holes [3]. Even new MAC protocols such as 802.16, designed with the benefit of hindsight, appear to have multiple security problems [4]. Likewise, there has been increased interest in MAC and PHY layers that either reduce power or contention for sensor networks and for *delay tolerant networks*. The 802.11 MAC layer is poorly suited for these novel network architectures. Most researchers in these domains limit their exploration to simulation. Sensor networking researchers do often "roll their own" MAC layers using simple radios interfaces and simplified protocols [5]. However, most of these protocol implementations and underlying network hardware platforms operate under power and bitrate constraints suitable for sensor networks. These constraints make them unsuited for experimentation in other interesting domains, *e.g.* mesh and community networking, long-distance networks and networks involving directional antennas.

At the same time, there has been a growing interest in *cognitive* or *software defined radios* [6, 7, 8], both from the perspective of the technology needed to make such systems a reality and from the possible channel management enabled by such systems. Software defined radios are key to providing the "spectrum agility" that would allow the current fixed allocation of spectrum to evolve to a more dynamic real-time allocation policy. Spectrum agility will require considerable coordination with the applications and operating systems of devices using particular frequencies – there are so many possible system configurations that managing such resources is very much an open problem. As SDR becomes a more common technology, protocols and behavioral norms will be needed to allow the use of new bandwidth without interfering with legacy services; no such systems currently exist.

We feel there is a compelling need for a *software defined radio* platform that is inexpensive, easy to use and could provide at least partial control over the MAC and PHY layers. Although the National Science Foundation is funding research programs charged with developing software defined radios, these systems are still expensive (typically costing $1,000-$10,000 for simple units and increasing to $100,000 for more complex systems) and relatively bulky when compared with the laptops and PDAs often used in wireless and mobile networking research. This greatly limits their utility when attempting to conduct large scale or mobile wireless *systems level* experiments. Furthermore, many interesting systems may still be constructed with a lesser degree of MAC and PHY flexibility than is afforded by these high-end platforms. Our own motivation was the need to develop MAC, link and routing protocols for wireless networks that involve *directionality*, either through the use of fixed directional antennas or through dynamic beam-steering antennas. We needed the ability to modify the standard 802.11 MAC protocol and to use multiple MAC protocols simultaneously.

We have developed the SOFTMAC system to fill this need. The SOFTMAC system uses a commodity 802.11a/b/g networking card with a chipset manufactured by the Atheros Corporation to build a software radio with predefined physical layers but a flexible MAC layer. Internally, the Atheros chipset provides considerable flexibility over the format of the transmitted packets, though this flexibility is not generally exposed by network drivers. By reverse-engineering many of those controls using a combination of open-source software references and "black box" probing and analysis, we have developed a driver that allows extensive control over the MAC layer while using any of the waveforms defined by the underlying 802.11b, 802.11g and 802.11a physical layers. We have also developed a software control system that allows us to run precise experiments, allowing us to address many of the "systems level" issues facing researchers interested in novel wireless networking applications.

In this paper, we discuss the highlights of our fully functional wireless networking experimentation system and how we are using it to solve problems when applying 802.11 networks [9, 10] to long-distance mesh networking. Unlike the work of Rao and Stocia [10] and others, we can avoid working around many of the problems of the 802.11 MAC, and can directly correct the performance problems that occur when applying that MAC to novel applications.

This software package has already been distributed to two companies and one additional university.

2 Implementation and Design of the SOFTMAC System

SOFTMAC provides precise control over the content and timing of wireless transmission and reception. Because the SOFTMAC system was implemented by overriding an implementation of the 802.11 MAC layer provided by a commercial family of networking cards, it is important to understand the key attributes of the 802.11 MAC and PHY layers and how they can help and hinder this overall goal: **a)** The PHY and MAC layers have checksums, and any failure in those checksums causes the message to be ignored; **b)** The MAC protocol is controlled by a series of precise timing intervals; **c)** Contention is handled by a combination of *carrier sensing* and *collision avoidance* using specified transmission durations contained in message headers; and **d)** the PLCP headers constitute a fixed overhead on each packet. At high data rate, this is about 15% of the transmission time of a packet.

Commodity 802.11 hardware typically divides up the functionality of the 802.11 MAC between the hardware/firmware on the card and the driver running on the host system. This means that the flexibility of such systems varies greatly between manufacturers. The SOFTMAC system relies on features of the chipsets designed by the Atheros Corporation. In particular, we have used the Atheros AR5212 chipsets and the open-source *Madwifi* driver [11]. Atheros uses a "hardware abstraction layer" or HAL to provide a common hardware interface for operating systems. The HAL is written in the machine code of the computer hosting the wireless card, and abstracts common functionality across different individual chipsets. Although the HAL is distributed in binary-only format and not extensively documented, there have been attempts to produce an "open-source" HAL. We have only used these open-source references while building SOFTMAC.

Overall, there were six primary tasks we needed to perform in order to implement SOFTMAC:

1. Override 802.11 MPDU frame format
2. Eliminate automatic ACK and retransmission
3. Eliminate RTS/CTS exchange

4. Eliminate virtual carrier sense (NAV)
5. Control PHY Clear Channel Assessment (CCA)
6. Control transmission backoff

The first three were done by operating the card in "Monitor Mode"; unlike many other cards, the Atheros chipset allows packets to be transmitted in monitor mode. By marking a packet as being a "retry" packet, changes normally made by the hardware are avoided. In monitor mode, the 802.11 NAV field is only applied when the destination address matches the actual MAC address; changing the MAC address or using an alternate format avoids throttling due to NAV. Measurements using an Agilent 4438C signal generator as a calibrated noise source shows that we can force transmission in monitor mode; furthermore, we can calibrate the local noise floor using a channel condition assessment mechanism provided by the Atheros chipset. Lastly, transmission backoff can be controlled by setting the specific contention register settings.

3 Evaluation and Microbenchmarks For SoftMac

In order to evaluate the performance and overhead of SOFTMAC, we developed a series of micro-benchmarks to measure important attributes of the system. These benchmarks were evaluated on a testbed of Shuttle SS51 small form-factor PC's equipped with 2.4 Ghz Celeron processors and a DLINK DWL-AG530 PCI card with external antenna. The DLINK cards use the Atheros AR5212 chipset.

An important performance characteristic is the rate at which packets may be sent and the delay between packets. By utilizing the high resolution timer built into the Atheros hardware we determined that it is possible to send messages every 91 ± 1 μseconds at the 95% confidence level.

Another important characteristic is the "turn-around" time for a received packet – this is the time between the completed reception of a packet and the ability to respond to information contained in that packet. This is a critical time constant for protocols that rely on an ACK mechanism, like the 802.11 protocol. We directly measured this time by having one station send a "request" packet that would solicit an immediate reply from a second station. A third station nearby monitored the traffic produced by both of these stations. The Atheros hardware timestamps each arriving packet with its own clock. By observing the differences of arrival times of each packet on the monitor machine and subtracting the time required to transmit the packet we obtained the interval between the end of the request packet and the start of the reply packet, *i.e.* the turnaround time. We used a data payload of 48 bytes sent at 1 Mb/s. Combined with the short preamble and four byte 802.11 MAC CRC this results in a transmission time of 512 microseconds. Over 100 trials we measured a mean turnaround time of 166 μseconds ± 1 μseconds at the 95% confidence level. As mentioned previously, by using the same destination address format as the 802.11 MAC, it's possible to have the hardware generate an ACK within 10 μseconds; however, that ACK is only generated when the standard CRC matches the computed checksum, and is only suitable for protocols that don't rely on more advanced error correction.

Observed Mean Inter-Event Spacing (in ms)		
Target	Mean	95% Conf. Int.
50 ms	49.96	± 0.12
1 ms	1.03	± 0.082
0.1 ms	0.091	± 0.001

Table 1: Inter-Event Timing Accuracy

It is important that the SOFTMAC system provides credible system-level performance. We implemented a simple MAC protocol that we called "Ethernet-over-wireless" (EoW); we encapsulated standard Ethernet packets as the PDSU payload. In many ways, this MAC protocol is similar to the 802.11 MAC, but it only uses two addresses (source & destination), has no NAV mechanism, does not use RTS/CTS and does not use an ACK mechanism. That protocol had essentially the same performance (throughput, latency) as the standard 802.11 protocol as implemented by the standard drivers in our test environment.

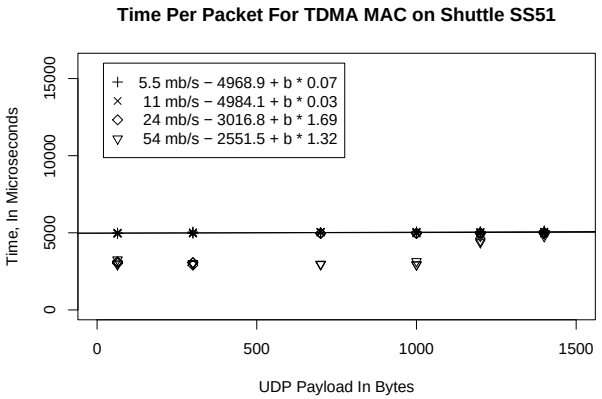
By putting the 802.11 cards into "monitor mode", it is possible that the interrupt load would increase dramatically. We used the Tracing Tools (LTT) to measure the number of interrupts per second as a function of the number of frames received and derived a linear regression model with coefficient of 1.2 interrupts per frame (with reasonably tight confidence bounds). The normal Linux timer interrupt causes 1000 interrupts per second, indicating that such an interrupt load is considered acceptable; we never saw more than 1020 interrupts per second for the network card, and felt that the low interrupt load was not worth moving to a polling driver. We also wanted to find the repeatability of remotely scheduled events (*e.g.* transmitting packets at scheduled times); any variance in the driver or scheduling issues in the operating system could limit the effectiveness of these events for timing sensitive experiments. We specified a 50 ms, 1 ms and 100 μs inter-event spacing. In this configuration, a control packet is scheduled every 3 seconds that resets the SOFTMAC device driver's time base. Following the control packet, data packets carrying the desired timing information are sent to the device at the specified intervals. This combination of control and data packets were measured for 50 trials. The resulting mean and 95% confidence interval for each inter-event spacing is shown in Table 1; there is little variance in the inter-packet timing. The low variance

for inter-packet timing implies that protocols could expect to rely on 100 μsecond inter-packet intervals – for example, the using SOFTMAC to build a "well known interfering source" could rely on tight bounds on inter-packet transmission times.

4 Using Multiple MAC Layers

In order to demonstrate the utility of SOFTMAC, we wanted to implement two MAC protocols that were distinctly different than the standard 802.11 MAC and that highlighted some aspect of the SOFTMAC system. We developed two simple MAC protocols; the first is a time-division multiplexing MAC with fixed duration slot times. The second is a CSMA/CA MAC protocol that used Reed-Solomon coding to reduce the bit-error rate at the expense of extra coding and CPU overhead. We also developed MULTIMAC, a system that lets researchers combine multiple MAC implementations. MULTIMAC acts as a wrapper to allow researchers to experiment with "cognitive radio" algorithms – it allows multiple lower-layer MAC protocols to be used simultaneously.

Our goal in implementing and describing these MAC protocols is to demonstrate the degree of control and the benefit that networking researchers can achieve from using SOFTMAC. Most current wireless networking research uses simulations; we think the community should start to insist on prototypes using systems such as SOFTMAC.



Values for 1 Mb/s rate					
Bytes	64	300	500	505	510
Time	5079.0	5515.0	6547.8	7079.5	7064.2

Values for 2 Mb/s rate				
Bytes	64	300	700	1000
Time	4972.33	5081.7	5124.1	5479.0

Figure 1: Regression model and means for time per packet using the "Ethernet over Wireless" MAC protocol on a Shuttle SS51 system with a 2.4 Ghz Celeron processor using the "TDMA" MAC and a 5 millisecond slot time. The data for the 1 Mb/s and 2 Mb/s rates are broken out because not all message sizes could be sent at those datarates with the specified slot time.

4.1 TDMA MAC Protocol

The 802.11 MAC provides media access using a collision-avoidance protocol that combines precise timing intervals and media sensing. There are a number of other MAC protocols are more efficient in certain scenarios; for example, the 802.16 [12] protocol uses a time-division multiplexing (TDMA) protocol where an access point downloads packets to individual stations using precise time slots. Variants of 802.16 can combine this with multiple channels to provide a continuous down-stream data link with an up-stream data-stream that may use another MAC protocol.

TDMA MAC protocols have the benefit of reducing packet scheduling variance and can be useful for multimedia applications; they can also be useful for "long haul" networks where carrier sensing may cause undesirable delays (which is one of our applications). To demonstrate the flexibility of the SOFTMAC system, we implemented a simple TDMA-based protocol. The protocol assigns fixed slot periods to individual stations; stations can only transmit during their slot period. Currently, the slot periods are assigned explicitly and there is no negotiation. The slot period is adjustable and the protocol is configurable such that stations can transmit either a single packet during their period or multiple packets. Timing for the slot period is done using a simple timing negotiation protocol using the precise timestamp that can be added to packets using the Atheros chipset.

The TDMA protocol was implemented as a kernel driver using the SOFTMAC interface. This was done to provide precise scheduling control of when packets could be transmitted.

Figure 1 shows the time-per-packet when using the TDMA protocol to communicate between two Shuttle SS51 computers using the same benchmark tools described earlier. The results are displayed differently for the different physical waveforms. The results for the 5.5, 11, 24 and 54 Mb/s waveforms are shown in the figure; the results for the the 1 and 2 Mb/s waveforms are shown in the table.

The first thing to note is that the time-per-packet is almost a constant 5 ms for the higher-speed waveforms, independent of the payload size. Regression models are shown for the higher speed waveforms, but the models are only very accurate for the 5.5 and 11 Mb/s waveforms. The constant time per packet is a consequence of using a TDMA protocol that allows a single message per slot time. The time is slightly lower for the higher speed protocols – this occurs because the measurement records the round-trip time for a

packet exchange between two nodes. For small message sizes, the higher speed protocols can send a message at the end of the timeslot of one node and have the second node finish replying near the beginning of its own time slot; thus, the elapsed time is less than 10 ms for an exchange. For the 5.5 and 11 Mb/s encoding, the transmission duration is longer, and the total time is closer to the total 10 ms time allocation. The 1 and 2 Mb/s waveforms are unable to transmit all message sizes in less than 5 ms, and the measurements that could be made for those waveforms are shown in the tables below the previous graph. As the packet length approaches 500 bytes for the 1mb/s rate, the different stations begin to lose their timeslots because they are still receiving messages from the other system when their time. This increases the overall packet delay as the system saturates. We found the 1mb/s waveform started to exhibit increased response time after ≈500 byte messages and the 2mb/s waveform had similar growth for ≈1000 byte messages, as predicted by a regression model of the time to transmit messages at different rates.

4.2 Adaptive Reed-Solomon MAC Protocol

To further demonstrate the feasibility of developing MAC protocols using SOFTMAC we designed an adaptive Reed-Solomon MAC which utilizes Reed-Solomon (RS) forward error correction to detect and fix bit errors in the MAC data payload. RS codes are a well known method of encoding data for protection against transmission errors. In the RS MAC, the common (255,223) encoding scheme is used. Because of the additional space and computational overhead associated with RS encoding, the MAC is adaptive and only uses forward error correction if bit errors are currently occurring. To simplify the development of this and other SOFTMAC-based MAC's, we augmented the Click Modular Router with a set of SOFTMAC components. This MAC was implemented as a Click application using the SOFTMAC Click elements and a standard RS software package. The CSMA/CA mechanism provided by SOFTMAC was used for channel access.

The RS MAC operates in one of two modes; all outgoing packets are RS encoded or all packets are sent unencoded. Since an endpoint cannot determine whether or not a packet it transmitted was received without error, it must rely on feedback from its peer to determine its current mode. A simple algorithm with three configurable parameters governs the sending of these feedback packets. The parameters are the sample period s, the error threshold e, and the no-error threshold c. Packets are observed over a sample period of s packets. If an endpoint is receiving unencoded packets and e or more packets with errors are received during a sample period, a packet is sent indicating that RS encoding should be used. Similarly, if an endpoint is currently receiving RS encoded packets and c or more packets are received without errors during the sample period, the MAC sends a message telling its peer to stop encoding packets. In unencoded packets, errors are detected using the CRC32 checksum that is computed and appended to the packet by the Atheros card. In RS encoded packets, errors are detected during the RS decoding process.

Reed-Solomon MAC for $s = 10$, $e = 2$, $c = 10$				
	Recv	Valid Recv	RS Recv	Corrections
R-S MAC	3859	3660	2971	23013
802.11	3845	1850	0	0

Figure 2: Packets received, Packets correctly received, Reed-Solomon packets received and Number of Reed-Solomon corrected bytes. Averages for 10 trials of 4000 packets each.

To test the functionality and performance of the adaptive Reed-Solomon MAC, we performed an experiment where two nodes try to send 1000 byte packets to each other at a rate of 100 packets per second. To decrease the probability of errors occurring in control frames relative to the probability of errors occurring in data frames, a data rate of 1 Mb/s was used for control information while data was sent at a rate of 54 Mb/s. Nodes were placed far enough apart to induce significant error when using the 54 Mb/s waveform. The result of 10 trials are shown in figure 2. For each test, 2000 packets were sent by each node for a total of 4000 packets. It is obvious from the results that the adaptive RS encoding scheme reduces the transmission error rate. On average, about 75% of packets were RS encoded, reducing the number of packets dropped due to errors from greater than 50% to less than 10%. The results also suggest that most errors occur in the 54 Mb/s payload portion of the packet and not in the 1 Mb/s and 2 Mb/s PLCP header. Errors were observed in more than half of the packets received and, as stated earlier, the PLCP header (which cannot be disabled in SOFTMAC) accounts for about 15% of the transmission time of a large packet for high data rates.

5 Related Work

There are other projects designed to facilitate MAC layer experimentation. In particular, an increasing amount of work is being done to construct Software Defined Radio platforms [13, 14, 15]. While these platforms are more flexible than SOFTMAC, they are also not "off the shelf" components, and are furthermore heavier, more expensive, and require more skill to assemble.

SOFTMAC bears some resemblance in functionality to 802.11 packet injection libraries [16]. However, the intended purpose of such libraries is typically related to security aspects of 802.11 networks rather than attempting to

create new MAC protocols.

A specific project that attempts to override the default behavior of 802.11 hardware in order to create a new MAC layer is the 2-P/SynOp MAC [17], created as a part of the Digital Gangetic Plains Project. However, this work attempts to utilize 802.11 hardware for a specific MAC layer, rather than creating a general purpose framework for enabling MAC layer experimentation.

The *Madwifi Stripped* [18] variant of the *Madwifi* [11] driver created as part of the MIT Roofnet project [19] allows 802.11 packet injection from the Click Modular Router. This driver is similar to SOFTMAC in that it affords a larger degree of control over 802.11 packet content and transmission parameters, but its focus appears to be centered more on directly enhancing 802.11-based networks rather than utilizing 802.11 hardware for broader MAC-layer experiments.

6 Implications of Cheap Software Defined Radios

In this paper, we have described a software system that lets networking researchers experiment with some aspects of software defined radio. We demonstrated that tight control over the inexpensive radio platform is possible (via the TDMA MAC), and that it can be used to implement features decidedly *not* implemented in standard wireless MAC protocols (via the Reed-Solomon MAC).

Although this platform is a pale imitation of a true software defined radio, the price ($\approx$ \$70) for the network card and the simple interface we have developed allows rapid experimentation. These tools should enable wireless researchers to validate their solutions in implementation.

In the future, we plan on augmenting the existing software to use more expensive (but more capable) purely software radios based on hybrid FPGA's that can deviate from the 802.11b/g PHY layers. The current SOFTMAC system can be downloaded from `http://systems.cs.colorado.edu/projects/softmac`.

This work was funded by NSF NeTS Prowin award #0435452 and #0435297 as well as an NSF RI award and NSF CRI Award #0454404.

References

[1] Phil Karn. Maca—a new channel access method for packet radio. In *Proceedings of 9th Amateur Radio Computer Networking Conference*, 1990. Also available at `http://www.ka9q.net/papers/maca.html`.

[2] J. Bellardo and S. Savage. 802.11 denial-of-service attacks: Real vulnerabilities and practical solutions. In *Proceedings of USENIX Security Symposium*, August 2003.

[3] Changhua He and John C. Mitchell. Analysis of the 802.11i 4-way handshake. In *WiSe '04: Proceedings of the 2004 ACM workshop on Wireless security*, pages 43–50. ACM Press, 2004.

[4] Derrick D. Boom. Denial of service vulnerabilities in IEEE 802.16 networks. Master's thesis, Naval Postgraduate School, Monterey, CA, Sept 2004. Available as `http://www.ieee802.org/16/tge/contrib/C80216e-04_406.pdf`.

[5] J.Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *The 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys'04)*, 2004.

[6] Joe Mitola. The software radio architecture. *IEEE Communications Magazine*, 33(5):26–38, May 1995.

[7] David L. Tennenhouse and Vanu G. Bose. Spectrumware: A software-oriented approach to wireless signal processing. In *Mobile Computing and Networking*, pages 37–47, 1995.

[8] V. Bose, M. Ismert, M. Wellborn, and J. Guttag. Virtual radios. *IEEE JSAC*, 17(4):591–602, April 1999.

[9] Joseph Dunn, Michael Neufeld, Anmol Sheth, Dirk Grunwald, and John Bennett. A practical cross-layer mechanism for fairness in 802.11 networks. In *Proceedings BROADNETS 2004*, pages 355–364, Oct 2004.

[10] Ananth Rao and Ion Stoica. An overlay MAC layer for 802.11 networks. In *MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 135–148, New York, NY, USA, 2005. ACM Press.

[11] Madwifi. `http://sourceforge.net/projects/madwifi`.

[12] Carl Eklund, Roger B. Marks, Kenneth L. Stanwood, and Stanley Wang. IEEE standard 802.16: A technical overview of the wirelessman air interface for broadband wireless access. `http://grouper.ieee.org/groups/802/16/tutorial`, Feb 2004.

[13] Gary Minden. Kansas university agile radio. (Private Communication), Sept 2005.

[14] Matt Ettus and Eric Blossom. The gnu software radio. Information at `http://www.ettus.com`, April 2005.

[15] D. Raychaudhuri *et al*. Network centric cognitive radio platform. Information at `http://www.winlab.rutgers.edu/pub/docs/focus/Cognitive-Hw.html`, 2005.

[16] lorcon - Loss Of Radio CONnectivity. `http://www.felloffthebackofatruck.com/svn/tx-80211`.

[17] Bhaskaran Raman and Kameswari Chebrolu. Revisiting MAC design for an 802.11-based mesh network. In *Second Workshop on Hot Topics in Networks (HOTNETS-III)*, 2004.

[18] Madwifi stripped. `http://www.pdos.lcs.mit.edu/~jbicket/madwifi.stripped`.

[19] Roofnet. `http://www.pdos.lcs.mit.edu/roofnet`.